(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。Int. Cl.[7]
G06F 9/30

(11) 공개번호 10-2005-0048465
(43) 공개일자 2005년05월24일

(21) 출원번호 10-2004-0083047
(22) 출원일자 2004년10월18일

(30) 우선권주장 10/715,688 2003년11월18일 미국(US)

(71) 출원인 인터내셔널 비지네스 머신즈 코포레이션
미국 10504 뉴욕주 아몬크 뉴오차드 로드

(72) 발명자 샌든피터에이
미국 05452 버몬트주 에섹스 정션 랭 드라이브 4
웨스트알마이클피
미국 05446 버몬트주 콜체스터 히든 오크스 드라이브 236

(74) 대리인 김창세
장성구
김원준

심사청구 : 있음

(54) 프로세서 및 매트릭스 데이터 처리 방법

요약

본 발명은 매트릭스 데이터를 처리하는 프로세서 및 방법에 관한 것이다. 프로세서는 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하도록 구성되는 M 개의 독립적인 벡터 레지스터 파일을 포함한다. 각각의 데이터 요소는 B 개의 이진 비트를 갖는다. 매트릭스는 N 개의 로우와 M 개의 컬럼을 가지며, L=N*M이다. 각각의 컬럼은 K 개의 서브컬럼을 갖는다. N≥2, M≥2, K≥1, B≥1이다. 매트릭스는 어레이 세트를 포함하며, 각각의 어레이는 그 매트릭스의 로우 또는 서브컬럼이다. 프로세서는 어레이 세트의 제 1 어레이에 대한 연산을 수행하는 인스트럭션을 실행할 수도 있으며, 상기 연산은 제 1 어레이의 데이터 요소에 대한 선택도로 수행된다.

대표도

도 2

명세서

도면의 간단한 설명

도 1은 본 발명의 실시예에 따른 데이터 요소들의 매트릭스의 레이아웃을 도시한 도면.

도 2는 본 발명의 실시예에 따른, 도 1의 매트릭스의 데이터 요소를 저장하기 위한 물리적인 레이아웃 및 데이터 요소를 도 1의 매트릭스로 판독하기 위한 멀티플렉서를 도시한 도면.

도 3은 본 발명의 실시예에 따른, 도 2의 물리적 레이아웃으로부터 도 1의 매트릭스의 로우 및 서브컬럼으로 데이터 요소를 판독하기 위한 판독 논리 테이블을 도시한 도면.

도 4는 본 발명의 실시예에 따른, 도 1의 매트릭스의 데이터 요소를 저장하기 위한 도 2의 물리적 레이아웃 및 도 1의 매트릭스의 데이터 요소를 물리적 레이아웃으로 기록하기 위한 멀티플렉서를 도시한 도면.

도 5는 본 발명의 실시예에 따른, 도 1의 매트릭스의 로우 및 서브컬럼으로부터 도 4의 물리적 레이아웃으로 데이터 요소를 기록하기 위한 기록 논리 테이블을 도시한 도면.

도 6(a) 내지 6(c)는 본 발명에 따른, 도 2 또는 도 4의 멀티플렉서를 이용하여 도 1의 매트릭스의 로우 또는 서브컬럼의 데이터 요소에 대한 선택도와 관련된 연산을 수행하는 인스트럭션을 도시한 도면.

도 7은 본 발명에 따른, 벡터 처리에 사용된 매트릭스의 로우 및 서브컬럼을 어드레스하기 위한 프로세서를 갖는 컴퓨터 시스템을 도시한 도면.

도면의 주요부분에 대한 부호의 설명

91 : 프로세서

92 : 입력 장치

93 : 출력 장치

94, 95 : 메모리 장치

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 프로세서 내의 복수의 벡터 레지스터 파일에 저장된 매트릭스의 로우 및 서브컬럼 모두를 논리적으로 어드레싱하는 것에 관한 것이다.

발명이 이루고자 하는 기술적 과제

벡터 및 매트릭스 계산과 관련된 연산을 위해 SIMD(Single Instruction Multiple Data) 벡터 처리 환경이 이용될 수도 있다. 이러한 계산 처리는 그래픽 및 디지털 비디오와 같은 다양한 멀티미디어와 관련될 수도 있다. 현재 SIMD 벡터 처리와 관련된 문제점은 벡터 데이터를 유연하게 처리할 필요가 있다는 데서 비롯된다. 벡터 데이터는 현재 표준 SIMD 계산으로 연산될 때 다수의 요소들의 단일(수평) 벡터로서 처리된다. 따라서 매트릭스의 로우는 종래의 방식에서 수평으로 액세스될 수 있다. 그러나, 흔히 매트릭스의 컬럼을 개체(eneity)로서 액세스할 필요가 있는데, 이것은 현재의 기법을 수행하는데 문제가 된다. 예를 들면, 매트릭스의 컬럼에 액세스하기 위해 매트릭스의 전치(transpose)를 발생하는 것이 일반적인데, 이것은 많은 수의 이동(move)/복사(copy) 인스트럭션을 요구하는 문제를 가지며, 요구된 레지스터의 수를 증가시킨다(즉, 적어도 두 배).

따라서, SIMD 벡터 처리에 사용된 매트릭스의 로우 및 컬럼을 어드레싱하기 위한 효과적인 프로세서 및 방법이 요구된다.

발명의 구성 및 작용

본 발명은 M 개의 독립적인 벡터 레지스터 파일을 포함하는 프로세서를 제공하는데, 이 M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하도록 구성되고, 각각의 데이터 요소는 B 개의 2진 비트를 가지며, 상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 가지며, L=N*M이고, 각각의 컬럼은 K 개의 서브컬럼을 가지며, N≥2, M≥2, K≥1, B≥1이며, N 개의 로우는 각각 어드레스가능하고, K 개의 서브컬럼은 각각 어드레스가능하며, 프로세서는 L 개의 데이터 요소를 중복 저장하지 않도록 구성된다.

본 발명은 매트릭스 데이터를 처리하기 위한 방법에 있어서, 프로세서를 제공하는 단계와, 프로세서 내에 M 개의 독립적인 벡터 레지스터 파일을 제공하는 단계 -M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하도록 구성되고, 각각의 데이터 요소는 B 개의 2진 비트를 가지며, 상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 가지며, L=N*M이고, 각각의 컬럼은 K 개의 서브컬럼을 가지며, N≥2, M≥2, K≥1, B≥1이며, N 개의 로우는 각각 어드레스가능하고, K 개의 서브컬럼은 각각 어드레스가능하며, 프로세서는 L 개의 데이터 요소를 중복 저장하지 않음- 를 포함하는 매트릭스 데이터 처리 방법을 제공한다.

본 발명은 M 개의 독립적인 벡터 레지스터 파일을 포함하는 프로세서에 있어서, M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하도록 구성되고, 각각의 데이터 요소는 B 개의 2진 비트를 가지며, 상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 가지며, L=N*M이고, 각각의 컬럼은 K 개의 서브컬럼을 가지며, N≥2, M≥2, K≥1, B≥1이며, N 개의 로우는 각각 어드레스가능하고, K 개의 서브컬럼은 각각 어드레스가능하며, 상기 매트릭스는 각각의 어레이가 매트릭스의 로우 또는 서브컬럼인 어레이 세트를 포함하고, 상기 프로세서는 어레이 세트의 제 1 어레이에 대한 연산을 수행하는 인스트럭션을 실행하도록 구성되며, 상기 연산은 제 1 어레이의 데이터 요소에 대한 선택도로 수행되는 프로세서를 제공한다.

본 발명은 매트릭스 데이터를 처리하기 위한 방법에 있어서, 프로세서를 제공하는 단계와, 프로세서 내에 M 개의 독립적인 벡터 레지스터 파일을 제공하는 단계 -M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하도록 구성되고, 각각의 데이터 요소는 B 개의 2진 비트를 가지며, 상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 가지며, L=N*M이고, 각각의 컬럼은 K 개의 서브컬럼을 가지며, N≥2, M≥2, K≥1, B≥1이며, N 개의 로우는 각각 어드레스가능하고, K 개의 서브컬럼은 각각 어드레스가능하며, 상기 매트릭스는 각각의 어레이가 매트릭스의 로우 또는 서브컬럼

인 어레이 세트를 포함함-와, 상기 프로세서에 의해 인스트럭션을 실행하는 단계 -상기 인스트럭션은 어레이 세트의 제 1 어레이에 대한 연산을 수행하고, 상기 연산은 상기 제 1 어레이의 데이터 요소에 대한 선택도로 수행됨- 를 포함하는 매트릭스 데이터 처리 방법을 제공한다.

본 발명은 바람직하게는 SIMD 벡터 처리에 사용된 매트릭스의 로우 및 컬럼을 어드레싱하기 위한 효과적인 프로세서 및 방법을 제공한다.

도 1은 본 발명의 실시예에 따른 데이터 요소의 매트릭스(10)의 레이아웃을 도시한 것이다. 매트릭스(10)는 128 개의 로우(로우 0, 1, ...,127로 표시됨)와 4 개의 컬럼(컬럼 0, 1, 2, 3으로 표시됨)을 포함한다. 로우 0, 1, ..., 127은 레지스터(R0, R1, .., R127)(즉, 레지스터 Rn, n=0, 1, ..., 127)로서 각각 어드레스된다. 컬럼은 다음과 같이 각각 서브컬럼으로 분할된다.

컬럼 0은 서브컬럼 128, 132, ..., 252로 분할된다.

컬럼 1은 서브컬럼 129, 133, ..., 253으로 분할된다.

컬럼 2는 서브컬럼 130, 134, ...,254로 분할된다.

컬럼 3은 서브컬럼 131, 135, ..., 255로 분할된다.

서브컬럼(128, 129, ..., 255)은 각각 레지스터(R128, R129, ..., R255)(즉, 레지스터 Rn, n=128, 129, ..., 255)로서 어드레스된다.

도 1은 또한 매트릭스(10)의 데이터 요소를 도시하고 있다. 각각의 데이터 요소는 B 개의 이진 비트(예를 들면, B=32)를 포함한다. 매트릭스(10)의 데이터 요소는 Rn[m] 형태를 갖는데, 여기서 n은 로우 인덱스(n=0, 1, ..., 127)이고, m은 컬럼 인덱스(m=0, 1, 2, 3)이다. 예를 들면, R5[2]는 매트릭스(10)의 5행 2열에 있는 데이터 요소를 나타낸다. 도 1에 도시된 바와 같이,

레지스터 R0은 로우 0(즉, 데이터 요소 R0[0], R0[1], R0[2], R0[3])을 포함하고,

레지스터 R1은 로우 1(즉, 데이터 요소 R1[0], R1[1], R1[2], R1[3])을 포함하고,

....

레지스터 R127은 로우 127(즉, 데이터 요소 R127[0], R127[1], R127[2], R1[3])을 포함하고,

레지스터 R128은 서브컬럼 0(즉, 데이터 요소 R0[0], R1[0], R2[0], R3[0])을 포함하고,

레지스터 R129는 서브컬럼 1(즉, 데이터 요소 R0[1], R1[1], R2[1], R3[1])을 포함하고,

...

레지스터 R255는 서브컬럼 128(즉, 데이터 요소 R0[128], R1[128], R2[128], R3[128])을 포함한다.

도 1의 매트릭스(10)의 데이터를 이동시켜 재편성하기 위한 인스트럭션은, 벡터 레지스터 파일, 벡터 레지스터 파일에 액세스하는 어드레스 레지스터 및 멀티플렉서를 포함하는 프로세서에 의해 처리된다. 따라서, 도 2는 본 발명의 실시예에 따라서, 벡터 레지스터 파일(V0, V1, V2, V3), 어드레스 레지스터(A0, A1, A2, A3) 및 4:1 멀티플렉서(m0, m1, m2, m3)를 나타낸다. 도 2에서, 벡터 레지스터 파일로부터 도 1의 매트릭스(10)의 로우 또는 서브컬럼을 판독하기 위해, 벡터 레지스터 파일이 어드레스 레지스터 및 멀티플렉서와 함께 사용된다. 각각의 벡터 레지스터 파일은 128 개의 레지스터를 포함한다. 상기 벡터 레지스터 파일의 수(4)는 도 1의 매트릭스(10)의 컬럼의 수(4)와 동일하다. 벡터 레지스터 파일 Vj(j=0, 1, 2, 3)는 레지스터 Yi[j](i=0, 1, ..., 127)(즉, Y0[j], Y1[j], ..., Y127[j])를 포함한다. 예를 들면, 벡터 레지스터 파일 V3(즉, j=3)은 레지스터 Y0[3], Y1[3], ..., Y127[3]을 포함한다. 각각의 레지스터 파일 V0, V1, V2, V3(및 그 내부의 128 개의 레지스터)은 어드레스 레지스터 A0, A1, A2, A3을 통해 각각 독립적으로 어드레스 가능하다. 일반적으로, 어드레스 레지스터 Aj(j=0, 1, 2, 3)는 Aj가 i(i=0, 1, ..., 127)를 포함하는 경우에 벡터 레지스터 파일의 어드레스 레지스터 Yi[j]를 어드레스한다. 예를 들면, 어드레스 레지스터 A2가 정수 4를 포함하면, 어드레스 레지스터 A2는 벡터 레지스터 파일 V2의 레지스터 Y4[2]를 어드레스한다.

도 1의 매트릭스(10)의 데이터 요소 Rn[m]은 도 2에 도시된 바와 같이 벡터 레지스터 파일(V0, V1, V2, V3) 내에 저장되어 분포된다. 도 2에서, 벡터 레지스터 파일(V0, V1, V2, V3)의 레지스터 내에 데이터 어레이 요소 Rn[m]를 분포시키면, 벡터 판독 동작 동안 도 1의 매트릭스(10)의 로우 및 서브컬럼 모두의 어드레싱이 용이해지는데, 이에 대해서는 아래에 도 3과 관련하여 설명한다. 도 2로부터 알 수 있듯이, 도 1의 매트릭스(10)는 다음 두 규칙에 따라서 벡터 레지스터 파일(V0, V1, V2, V3)에 저장된다.

제 1 규칙은 매트릭스(10)의 로우를 벡터 레지스터 파일에 저장하는 것과 관련이 있다. 제 1 규칙은 다음과 같다. 즉, 만약 데이터 요소 Rn[m]가 레지스터 Yn[j]에 저장되면, 데이터 요소 R(n)[m1]은 레지스터 Y(n)[j1]에 저장되는데, 여기서 j1=(j+ 1)mod4(즉, j=0, 1, 2, 3이 j1=1, 2, 3, 0으로 각각 맵핑됨)이고, m1=(m+ 1)mod4(즉, m=0, 1, 2, 3이 m1=1, 2, 3, 0으로 각각 맵핑됨)이다. 연산자 "mod"는 다음과 같이 정의된 모듈러스(modulus) 연산자이다. 만약 I1 및 I2가 정의 정수

이면, I1 mod I2는 I1을 I2로 나눈 나머지이다. 제 1 규칙의 예로서, 레지스터 R0과 관련된 로우의 데이터 요소 R0[0], R0[1], R0[2], R0[3]는 각각 레지스터 Y0[0], Y0[1], Y0[2], Y0[3]에 저장되고, 레지스터 R1과 관련된 로우의 데이터 요소 R1[0], R1[1], R1[2], R1[3]은 각각 레지스터 Y1[1], Y1[2], Y1[3], Y1[0]에 저장된다. 제 1 규칙의 결과로서, 로우 n의 각각의 데이터 요소 Rn[0], Rn[1], Rn[2], Rn[3]은 상이한 벡터 레지스터 파일에 저장되지만, 각각의 벡터 레지스터 파일 내의 동일한 상대적인 레지스터 위치(즉, 레지스터 Yi[j]에 대해 i=n)에 저장된다. 따라서, 레지스터 Rn과 관련된 로우의 데이터 요소 Rn[0], Rn[1], Rn[2], Rn[3]이 도 2의 레지스터 Yn[0], Yn[1], Yn[2], Yn[3] 내의 치환된 시퀀스(permuted sequence)로서 저장된다.

제 2 규칙은 매트릭스(10)의 서브컬럼을 벡터 레지스터 파일에 저장하는 것과 관련이 있다. 즉, 데이터 요소 Rn[m]이 레지스터 Yn[j]에 저장되면, 데이터 요소 R(n+ 1)[m]는 레지스터 Y(n+ 1)[j1]에 저장되는데, 여기서 j1=(j+ 1)mod4이다. 제 2 규칙의 일례로서, 레지스터(R129)에 의해 지시된 서브컬럼의 데이터 요소 R0[1], R1[1], R2[1], R3[1]이 각각 레지스터 Y0[1], Y1[2], Y2[3], Y3[0]에 저장된다. 제 2 규칙의 결과로서, 로우 n의 각각의 데이터 요소 Rn[0], Rn[1], Rn[2], Rn[3]은 상이한 벡터 레지스터 파일에 저장되며 각각의 벡터 레지스터 파일 내의 레지스터 Yi[j]에 대해 인덱스 i를 특징으로 하는 상이한 상대적인 벡터 레지스터 위치에 저장된다. 따라서, 각각의 서브컬럼의 데이터 요소는 벡터 레지스터 파일(V0, V1, V2, V3)의 레지스터 내에 불규칙한 대각선 형식(broken diagonal fashion)으로 저장된다.

도 2의 멀티플렉서(m0, m1, m2, m3)는 벡터 레지스터 파일(V0, V1, V2, V3)과 멀티플렉서(m0, m1, m2, m3) 사이의 논리 상호 접속부(17)와 함께 벡터 레지스터 파일(V0, V1, V2, V3)로부터 판독된 데이터 요소를 순차적으로 정렬시킨다. 논리 상호 접속부(17)는 도 3에 도시된 판독 논리 테이블(read-logic table)(20)에서 설명되는데, 이하에 논의된다.

도 3은 본 발명의 실시예에 따라, 도 2의 멀티플렉서(m0, m1, m2, m3)를 이용하여 벡터 레지스터 파일(V0, V1, V2, V3)로부터 도 1의 매트릭스(10)의 로우 및 서브컬럼을 판독하기 위한 판독 논리 테이블(20)을 도시하고 있다. 도 3에서, 판독 논리 테이블(20)의 컬럼 21은 도 1의 레지스터(R0, R1, ..., R255)를 리스트한다. 판독 논리 테이블(20)의 컬럼 22 내지 25는 어드레스 레지스터 A0, A1, A2, A3의 값을 리스트한다. 판독 논리 테이블(20)의 컬럼 26 내지 29는 멀티플렉서(m0, m1, m2, m3)의 값을 리스트한다. 각각의 상기 멀티플렉서(m0, m1, m2, m3)는 두 쌍의 스위치 세트이며, 각 스위치는 "온(on)" 또는 "오프(off)"이며 이진 비트 1 또는 0으로 각각 표현된다. 따라서, 멀티플렉서의 "값"은 두 스위치의 온/오프 상태를 각각 나타내는 두 개의 이진 비트의 합성 값(0, 1, 2 또는 3)이다.

판독되는 매트릭스(10)의 각각의 로우는 0≤n≤127 범위 내의 레지스터(Rn)를 선택하는 인덱스(n)에 의해 식별된다. 판독되는 매트릭스(10)의 각각의 서브컬럼은 128≤n≤255 범위 내의 레지스터(Rn)를 선택하는 인덱스(n)에 의해 식별된다. 판독되는 각각의 로우 또는 서브컬럼의 데이터 요소는 벡터 레지스터 파일(V0, V1, V2, V3)의 레지스터(Yi[j])로부터 액세스되며, 이 레지스터는 어드레스 레지스터(A0, A1, A2, A3)에 의해 각각 지시된다. 어드레스 레지스터(A0, A1, A2, A3)에 의해 지시된 레지스터로부터 액세스된 데이터 요소는 다음과 같이 멀티플렉서(m0, m1, m2, m3)의 값에 따라서 순차적으로 정렬된다. 멀티플렉서 값은 벡터 레지스터 파일(V0, VQ, V2 또는 V3)을 선택하는 인덱스(j)이다. 그 다음에 선택된 벡터 레지스터 파일과 관련된 어드레스 레지스터의 내용이 데이터 요소를 선택한다. Yi[j]는 벡터 레지스터 파일(Vj)의 레지스터(i)를 나타낸다는 것을 상기하라. 만약 판독될 로우 또는 서브컬럼이 레지스터(Rn)에 의해 식별되면, 데이터 요소는 Y(a0)[m0], Y(a1)[m1], Y(a2)[m2], Y(a3)[m3] 순서로 레지스터(Yi[j])로부터 액세스되는데, 여기서 a0, a1, a2, a3은 A(m0), A(m1), A(m2), A(m3)의 내용을 각각 나타낸다. 예를 들면, A0=2, A1=3, A2=0, A3=1이고, m0=3, m1=2, m2=1, m3=0이면,

a0=1(즉, A(m0) 또는 A3의 내용)이고,

a1=0(즉, A(m1) 또는 A2의 내용)이며,

a2=3(즉, A(m2) 또는 A1의 내용)이고,

a3=2(즉, A(m3) 또는 A0의 내용)이다.

로우 판독의 예로서, 판독될 로우가 레지스터 R2(도 1 참조)와 관련된다고 가정하자. 그러면, 도 3의 R2 로우로부터, A0=2, A1=2, A2=2, A3=2이며, m0=2, m1=3, m2=0, m3=1이다. 데이터 요소는 m0, m1, m2, m3의 값에 의해 각각 규정된 바와 같이 Y(a0)[2], Y(a1)[3], Y(a2)[0], Y(a3)[1]의 순서로 레지스터(Ri[j])로부터 액세스된다. A0, A1, A2, A3 및 m0, m1, m2, m3의 값을 사용하면, a0=2, a1=2, a2=2, a3=2가 된다. 따라서, 데이터 요소는 Y2[2], Y2[3], Y2[0], Y2[1]의 순서로 레지스터(Ri[j])로부터 액세스된다. 따라서, Yi[j]의 내용에 대하여 도 2를 참조하면, 데이터 요소는 R2[0], R2[1], R2[2], R2[3]의 순서로 액세스되는데, 이것은 레지스터 R2와 관련된 로우의 데이터 요소의 올바른 정렬로서 도 1로부터 확인할 수도 있다.

서브컬럼 판독의 일례로서, 판독되는 서브컬럼이 레지스터 R129(도 1 참조)와 관련된다고 가정하자. 도 3의 R129 로우로부터, A0=3, A1=0, A2=1, A3=2이며, m0=1, m1=2, m2=3, m3=0이다. 따라서, 데이터 요소는 m0, m1, m2, m3의 값에 의해 각각 규정된 바와 같이 Y(a0)[1], Y(a1)[2], Y(a2)[3], Y(a3)[0]의 순서로 레지스터(Y[j])로부터 액세스된다. A0, A1, A2, A3 및 m0, m1, m2, m3의 값을 사용하면, a0=0, a1=1, a2=2, a3=3이 된다. 따라서, 데이터 요소는 Y0[1], Y1[2], Y2[3], Y3[0]의 순서로 레지스터(Ri[j])로부터 액세스된다. 따라서, Yi[j]의 내용에 대하여 도 2를 참조하면, 데이터 요소는 R0[1], R1[1], R2[1], R3[1]의 순서로 액세스되는데, 이것은 레지스터 R129와 관련된 서브컬럼의 데이터 요소의 올바른 정렬로서 도 1로부터 확인할 수도 있다.

앞의 예는, 도 1의 매트릭스(10)의 로우 또는 서브컬럼을 정확하게 판독하도록 멀티플렉서(m0, m1, m2, m3)가 액세스된 데이터 요소를 순차적으로 정렬하기 위해, 벡터 레지스터 파일의 레지스터 내의 데이터 요소의 저장 장소를 고려하여 다음의 일반적인 규칙이 준수된다는 것을 나타낸다. 각각의 서브컬럼의 데이터 요소는 상이한 벡터 레지스터 파일에 저장되는데, 이것은 각각의 서브컬럼에 대하여, 그 내부의 두 데이터 요소가 동일한 벡터 레지스터 파일에 저장되지 않는다는 것을 의미한다. 마찬가지로, 각 로우의 각각의 데이터 요소는 상이한 벡터 레지스터 파일에 저장되는데, 이것은 각각의 로우에 대하여, 그 내부의 두 데이터 요소가 동일한 벡터 레지스터 파일에 저장되지 않는다는 것을 의미한다. 도 2는 벡터 레

지스터 파일(V0, V1, V2, V3)의 레지스터(Yi[j]) 내의 데이터 어레이 요소(Rn[m])의 특정 분포를 나타내지만, 본 발명의 범주 내에 상기 일반적인 규칙이 준수되는 데이터 어레이 요소(Rn[m])의 다른 분포가 있을 수도 있다. 로우 또는 서브컬럼을 판독하기 위한 판독 논리 테이블(예를 들면, 도 3 참조)은 레지스터(Yi[j]) 내의 데이터 어레이 요소(Rn[m])의 특정 분포에 특유하다.

따라서, 멀티플렉서(m0, m1, m2, m3)는 커맨드에 응답하여 도 3의 판독 논리 테이블(20)에 의해 예시된 바와 같이 판독-로우(또는 판독 컬럼) 맵핑에 따라서 벡터 레지스터 파일(V0, V1, V2, V3)로부터 로우(또는 서브컬럼)로 로우(또는 서브컬럼)의 데이터 요소를 맵핑시킴으로써, 매트릭스의 로우(또는 서브컬럼)를 판독하도록 구성된다. 숫자 값들을 갖는 판독 논리 테이블(20)을 사용하는 대신에, 불 논리 스테이트먼트(Boolean logic statement)의 사용에 의해 판독 로우(또는 판독 컬럼) 맵핑 알고리즘을 구현할 수 있다.

도 2는 전술한 바와 같이, 도 3의 판독 논리 테이블(20)에 따라서 레지스터(Yi[j])로부터 도 1의 매트릭스(10)의 로우 또는 서브컬럼을 판독하는 것과 관련이 있다. 다음에 설명하는 바와 같이, 도 4는 도 5의 기록 논리 테이블(write-logic table)(40)에 따라서 도 1의 매트릭스(10)의 로우 또는 서브컬럼을 레지스터(Yi[j])에 기록하는 것과 관련이 있다.

도 4는 본 발명의 실시예에 따른, 벡터 레지스터 파일(V0, V1, V2, V3), 어드레스 레지스터(A0, A1, A2, A3) 및 4:1 멀티플렉서(m0, m1, m2, m3)를 포함하는 프로세서(15)를 도시하고 있다. 도 4에서, 벡터 레지스터 파일은 벡터 레지스터 파일(V0, V1, V2, V3)에 도 1의 매트릭스(10)의 로우 또는 서브컬럼을 기록하기 위해 어드레스 레지스터 및 멀티플렉서와 함께 사용된다. 벡터 레지스터 파일(V0, V1, V2, V3), 어드레스 레지스터(A0, A1, A2, A3) 및 벡터 레지스터 파일의 레지스터(Yi[j]) 내의 도 1의 매트릭스(10)의 데이터 요소(Rn[m])의 분포는 위에서 설명한 도 2에서와 동일하다. 도 4에서, 벡터 레지스터 파일(V0, V1, V2, V3)의 레지스터 내의 데이터 어레이 요소(Rn[m])의 분포는 벡터 기록 동작 동안에 도 1의 매트릭스(10)의 로우 및 서브컬럼 모두의 어드레싱을 용이하게 하는데, 이에 대해서는 아래에서 도 5와 함께 설명한다.

도 4의 멀티플렉서(m0, m1, m2, m3)는 벡터 레지스터 파일(V0, V1, V2, V3)과 멀티플렉서(m0, m1, m2, m3) 사이의 논리 상호접속부(18)와 함께 벡터 레지스터 파일(V0, V1, V2, V3)에 기록될 데이터 요소를 순차적으로 정렬한다. 논리 상호 접속부(18)는 도 5에 도시된 기록 논리 테이블(40)에 도시되어 있으며, 다음에 논의한다.

도 5는 본 발명에 따른, 도 2의 멀티플렉서(m0, m1, m2, m3)를 이용하여, 도 1의 매트릭스의 로우 및 서브컬럼을 벡터 레지스터 파일(V0, V1, V2, V3)에 기록하기 위한 기록 논리 테이블(40)을 도시한 것이다. 도 5에서, 기록 논리 테이블(40)의 컬럼 41은 도 1의 레지스터(R0, R1, ..., R255)를 리스트한다. 기록 논리 테이블(40)의 컬럼(42 내지 45)은 어드레스 레지스터(A0, A1, A2, A3)의 값을 리스트한다. 기록 논리 테이블(40)의 컬럼(46 내지 49)은 멀티플렉서(m0, m1, m2, m3)의 값을 리스트한다. 기록될 매트릭스(10)의 각각의 로우는 0≤n≤127 범위 내의 레지스터(Rn)를 선택하는 인덱스 n에 의해 식별된다. 기록될 매트릭스(10)의 각각의 서브컬럼은 128≤n≤255 범위의 레지스터(Rn)를 선택하는 인덱스 n에 의해 식별된다.

레지스터 Rn(n=0, 1, ..., 255)에 의해 선택된, 각각의 로우 또는 서브컬럼의 데이터 요소는 다음의 규칙에 따라서 벡터 레지스터 파일(V0, V1, V2, V3)의 레지스터(Yi[j])에 분포된다. Yi[j]는 벡터 레지스터 파일(Vj)의 레지스터(i)를 나타낸다는 점을 상기하라. 레지스터 Rn(도 1에 나타낸 바와 같은)과 관련된 순차적으로 정렬된 데이터 요소가 Rn[0], Rn[1], Rn[2], Rn[3]으로 표시된다고 하자. 상기 규칙은, 데이터 요소(Rn[0], Rn[1], Rn[2], Rn[3])가 벡터 레지스터 파일(V(j0), V(j1), V(j2), V(j3))에 각각 기록되며, 여기서 멀티플렉서(m(j0), m(j1), m(j2), m(j3))는 0, 1, 2, 3을 각각 포함한다는 것이다. 일례로서, m0=1, m1=2, m2=3, m3=0이면, Rn[0], Rn[1], Rn[2], Rn[3]이 m3=0, m0=1, m1=2, m2=3을 반영하여 벡터 레지스터 파일 V3, V0, V1, V2에 각각 기록된다. 어드레스 레지스터(A0, A1, A2, A3)는 데이터 요소가 각각 기록되는 벡터 레지스터 파일(V0, V1, V2, V3) 내의 레지스터 번호를 포함한다. 따라서, 앞의 예에서, 어드레스 레지스터 A3이 값 34를 포함하면, 데이터 요소 Rn[0]은 벡터 레지스터 파일 V3의 레지스터 34에 기록된다.

로우를 기록하는 예로서, 기록되는 로우가 레지스터 R2(도 1 참조)와 관련된다고 가정하자. 도 1의 R2 로우로부터, R2와 관련된 데이터 요소의 순서는 R2[0], R2[1], R2[2], R2[3]이다. 도 5의 R2 로우로부터, A0=2, A1=2, A2=2, A3=2이고, m0=2, m1=3, m2=0, m3=1이다. 따라서, 이 규칙에 따르면, 레지스터(R2)와 관련된 데이터 요소(R2[0], R2[1], R2[2], R2[3])의 순서는 m2=0, m3=1, m0=2, m1=3을 반영하여 벡터 레지스터 파일 V2, V3, V0, V1에 분포된다. 따라서, A2=2가 도 4와 부합하기 때문에, 데이터 요소 R2[0]은 벡터 레지스터 파일 V2의 레지스터 위치 2(즉, Y2[2])에 기록된다. A3=2가 도 4와 부합하기 때문에, 데이터 요소 R2[1]은 벡터 레지스터 파일 V3의 레지스터 위치 2(즉, Y2[3])에 기록된다. A0=2가 도 4와 부합하기 때문에, 데이터 요소 R2[2]는 벡터 레지스터 파일 V0의 레지스터 위치 2(즉, Y2[0])에 기록된다. A1=2가 도 4와 부합하기 때문에, 데이터 요소 R2[3]은 벡터 레지스터 파일 V1의 레지스터 위치 2(즉, Y2[1])에 기록된다.

서브컬럼 기록의 예로서, 기록되는 서브컬럼이 레지스터 R129(도 1 참조)와 관련된다고 가정하자. 도 1의 R129로부터, R129와 관련된 데이터 요소의 순서는 R0[1], R1[1], R2[1], R3[1]이다. 도 5의 R129 로우로부터, A0=3, A1=0, A2=1, A3=2이고, m0=3, m1=0, m2=1, m3=2이다. 따라서, 상기 규칙에 따르면, 레지스터 R129와 관련된 데이터 요소(R0[1], R1[1], R2[1], R3[1])의 순서는 m1=0, m2=1, m3=2, m0=3을 반영하여 벡터 레지스터 파일(V1, V2, V3, V0)에 분포된다. 따라서, A1=0이 도 4와 부합하기 때문에, 데이터 요소 R0[1]은 벡터 레지스터 파일 V1의 레지스터 위치 0(즉, Y0[1])에 기록된다. A2=1이 도 4와 부합하기 때문에, 데이터 요소 R1[1]은 벡터 레지스터 파일 V2의 레지스터 위치 1(즉, Y1[2])에 기록된다. A3=2가 도 4와 부합하기 때문에, 데이터 요소 R2[1]은 벡터 레지스터 파일 V3의 레지스터 위치 2(즉, Y2[3])에 기록된다. A0=3이 도 4와 부합하기 때문에, 데이터 요소 R3[1]은 벡터 레지스터 파일 V0의 레지스터 위치 3(즉, Y3[0])에 기록된다.

따라서, 멀티플렉서(m0, m1, m2, m3)는 커맨드에 응답하여, 도 5의 기록 논리 테이블(40)에 의해 예시된 바와 같이 기록 로우(또는 기록 컬럼) 맵핑 알고리즘에 따라서 로우(또는 서브컬럼)의 데이터 요소를 벡터 레지스터 파일(V0, V1, V2, V3)에 맵핑시킴으로써 매트릭스의 로우(또는 서브컬럼)를 기록하도록 구성된다. 숫자 값들을 갖는 기록 논리 테이블(40)을 사용하는 대신에, 불 논리 스테이트먼트(Boolean logic statement)의 사용에 의해 기록 로우(또는 기록 컬럼) 맵핑 알고리즘을 구현할 수 있다.

도 1 내지 5에 도시된 실시예들은 128 개의 로우 및 4 개의 컬럼을 갖는 매트릭스(여기서, 각각의 컬럼은 32개의 서브컬럼으로 분할되고, 각각의 서브컬럼 내에 4개의 데이터 요소를 포함함)를 설명하였지만, 본 발명의 범위는 일반적으로 N 개의 로우와 M 개의 컬럼을 가지며 따라서 L=N*M인 총 L 개의 데이터 요소를 포함하는 매트릭스를 포함한다. N 개의 로우 각각 및 K 개의 서브컬럼 각각은 어드레스 가능하다. 각각의 데이터 요소는 B 개의 이진 비트를 포함한다. 파라미터 N, M, K, B는 조건 N≥2, M≥2, K≥1, B≥1을 따를 수도 있다. 도 1 내지 5에 도시된 예에서, N=128, M=4, K=32, B=32이다.

도 1 내지 5에 도시된 예는 N, M, K를 포함하는 다음의 관계, 즉 K*M=N, N mod K=0, N mod M=0, $N=2^P$(P는 적어도 2인 양의 정수), $M=2^Q$(Q는 적어도 2인 양의 정수)를 나타내며, 각 컬럼의 각각의 서브컬럼은 N 개의 로우 중 M 개의 로우를 포함하고, 각 서브컬럼 내의 총 이진 비트 수 및 각 로우 내의 총 이진 비트 수는 일정한 이진 비트 수(도 1 내지 5에서 128비트)와 동일하다.

N, M, K를 포함하는 상기 관계는 단순히 예시일 뿐 제한적인 것은 아니다. 다음의 대안적인 비제한적인 관계가 본 발명의 범주 내에 포함된다. 제 1 대안적인 관계는 소정의 컬럼의 서브컬럼이 동일한(즉, 일정한) 수의 데이터 요소를 갖지 않는다는 것이다. 제 2 대안적인 관계는 각각의 서브컬럼 내의 총 이진 비트 수가 각각의 로우 내의 총 이진 비트 수와 동일하지 않다는 것이다. 제 3 대안적인 관계는 적어도 두 개의 컬럼이 상이한 수(K)의 서브컬럼을 갖는다는 것이다. 제 4 대안적인 관계는 N mod K≠0이다. 제 5 대안적인 관계는 어떠한 P의 값도 $N=2^P$(P는 적어도 2인 정수)을 만족시키지 않는다는 것이다. 제 6 대안적인 관계는 어떠한 Q의 값도 $M=2^Q$(Q는 적어도 2인 양의 정수)을 만족시키지 않는다는 것이다.

본 발명의 범위는 또한 각각의 데이터의 B 개의 이진 비트가 부동 소수점 수, 정수, 비트 스트링 또는 문자 스트링을 나타내도록 구성되는 실시예를 포함한다.

또한, 본 발명은 복수의 벡터 레지스터 파일을 갖는 프로세서를 포함한다. 복수의 벡터 레지스터 파일은 L 개의 데이터 요소로 이루어진 매트릭스를 집합적으로 저장하도록 구성된다. L 개의 데이터 요소는 프로세서 내에 중복 저장되도록 요구되지 않는데, 이는 도 1 내지 5와 관련하여 위에서 설명한 바와 같이, 매트릭스의 로우 및 서브컬럼이 프로세서 내의 어드레스 레지스터 및 멀티플렉서와 함께 벡터 레지스터 파일을 사용하여 개별적으로 어드레스 가능하기 때문이다.

도 1 내지 5와 관련하여 위에서 설명한 본 발명의 실시예에서, 각각의 서브컬럼의 데이터 요소는 상이한 벡터 레지스터 파일에 저장되도록 구성되고, 각 로우의 데이터 요소는 상이한 벡터 레지스터 파일에 저장되도록 구성된다. 또한, 각각이 서브컬럼의 데이터 요소는 상이한 벡터 레지스터 파일의 상이한 상대적인 레지스터 위치에 저장되도록 구성되고, 각각의 로우의 데이터 요소는 상이한 벡터 레지스터 파일의 동일한 상대적인 레지스터 위치에 저장되도록 구성된다.

수평으로 배향된 N 개의 로우와 수직으로 배향된 M 개의 컬럼을 갖는 매트릭스(10)가 도 1에 도시되어 있지만, 본 발명의 범위는 N 개의 로우가 수직으로 배향되고 M 개의 컬럼이 수평으로 배향되는 실시예도 포함한다.

도 6(a) 내지 6(c)는 본 발명의 실시예에 따른, 도 2 또는 도 4의 멀티플렉서를 이용하여 도 1의 매트릭스의 로우 또는 서브컬럼의 데이터 요소에 대한 선택도(selectivity)로 연산을 수행하는 인스트럭션을 도시한 것이다.

도 6(a)는 레지스터 RA와 관련된 어레이 R(RA)의 데이터 요소가 레지스터 DEST와 관련된 어레이 R(DEST) 내의 데이터 요소 위치에 복사되는 인스트럭션을 도시한 것이다. 2비트 워드 aa, bb, cc 및 dd는 각각 도 2 또는 도 4의 멀티플렉서 m0, m1, m2, m3의 값에 각각 대응한다. 어레이 R(RA)가 그 내부에 데이터 요소 R(RA)[0], R(RA)[1], R(RA)[2], R(RA)[3]을 갖는다고 하자. 어레이 R(DEST)가 그 내부에 데이터 요소 R(DEST)[0], R(DEST)[1], R(DEST)[2], R(DEST)[3]을 갖는다고 하자. 도 6(a)의 동작은 R(RA)[aa], R(RA)[bb], R(RA)[cc], R(RA)[dd]를 R(DEST)[0], R(DEST)[1], R(DEST)[2], R(DEST)[3]으로 각각 복사한다. 따라서, 멀티플렉서 값 m0, m1, m2, m3은 어레이 R(RA)의 요소들에 대한 선택도로 어레이 R(RA)로부터 어레이 R(DEST)로의 데이터의 이동을 제어한다. 설명을 위해, 다음 세 가지 예를 고려한다.

도 6(a)에 도시된 인스트럭션에 대한 제 1 예에서, aa=0, bb=1, cc=2, dd=3으로 설정한다. 이것은 요소들 R(RA)[0], R(RA)[1], R(RA)[2], R(RA)[3]이 R(DEST)[0], R(DEST)[1], R(DEST)[2], R(DEST)[3]으로 각각 복사되는 종래의 어레이 복사 동작이다.

도 6(a)에 도시된 인스트럭션에 대한 제 2 예에서, aa=0, bb=0, cc=0, dd=0으로 설정하면, 결국 R(RA)[0]이 각각의 R(DEST)[0], R(DEST)[1], R(DEST)[2], R(DEST)[3]으로 복사된다. 이 기능은 흔히 'splat' 연산이라고도 하는데, 스칼라-벡터(scalar-vector) 연산을 지원한다.

도 6(a)에 도시된 인스트럭션에 대한 제 3 예에서, aa=3, bb=2, cc=1, dd=0으로 설정하면, 결국 R(RA)[3], R(RA)[2], R(RA)[1], R(RA)[0]이 R(DEST)[0], R(DEST)[1], R(DEST)[2], R(DEST)[3]으로 각각 복사된다. 따라서 R(RA)는 R(RA)의 데이터 요소의 역순으로 R(DEST)로 복사된다.

이들 예는 단순히 예시적일 뿐이다. aa, bb, cc, dd의 256 개의 순열(permutation)(즉, 4^4)이 있기 때문에, 도 6의 연산은 256 개의 연산 변수를 포함한다. 또한, RDEST≠RA 및 RDEST=RA가 모두 가능하다. 따라서, RDEST=RA의 경우가 256 개의 다양한 순열 중 어느 하나에 따라서 R(RA)의 데이터 요소를 내부적으로 재배열하는 것을 용이하게 한다. 이 모든 연산은 멀티플렉서(m0, m1, m2, m3)의 사용을 요구한다. 도 1 내지 5와 관련하여 앞서 설명한 바와 같이, 도 1의 매트릭스(10)의 로우 및 서브컬럼의 어드레싱을 달성하기 위해 멀티플렉서(m0, m1, m2, m3)가 존재해야 하므로, 이 모든 연산은 기본적으로 자유롭다.

도 6(b)는 R(RA)의 선택된 요소를 마스킹하고, 레지스터 RA와 관련된 어레이 R(RA)의 데이터 요소가 레지스터 DEST와 관련된 어레이 R(DEST) 내의 데이터 요소 위치에 복사되는 인스트럭션을 도시한 것이다. 즉, R(RA)의 Q 개의 요소가

R(DEST)로 마스킹되고(즉, 복사되지 않고), R(RA)의 나머지 4-Q 개의 요소가 R(DEST)로 복사되는데, 여기서 0≤Q≤4이다. B0, B1, B2 B3이 이 연산에 요구된 마스크 비트를 나타낸다고 하자. 그러면, m=0, 1, 2, 3에 대해 Bm=1/0인 경우에 R(RA)[m]은 R(DEST)로 복사되거나 복사되지 않는다. 이것은 일반적으로 판독-수정-기록 순서(read-modify-write sequence)로 수행되지만, 개별 벡터 레지스터 파일(V0, V1, V2, V3)의 사용에 의해 용이해진다.

도 6(c)는 레지스터 RA와 관련된 어레이 R(RA)의 단일 데이터 요소가 레지스터 RB와 관련된 어레이 R(RB)와 기능적으로(함수 f에 따라서) 결합되는 것을 도시하고 있다. 기능적인 결과는 레지스터 DEST와 관련된 어레이 R(DEST)에 저장되고, R(RA)[aa]의 요소는 함수 f를 수행하는데 사용된다. 2 비트 워드 aa는, 네 개의 모든 멀티플렉서가 단일 데이터 요소를 선택하도록 판독 멀티플렉서 m0, m1, m2, m3(도 2)를 설정함으로써 레지스터 RA와 관련된 어레이 R(RA)의 단일 데이터 요소를 선택한다. 예를 들면, 함수 f가 "덧셈(addition)"를 나타내면, 다음의 SUM 벡터(성분 SUM[0], SUM[1], SUM[2], SUM[3]을 갖는)가 형성되어 R(DEST)에 저장된다.

SUM[0]=R(RA)[aa]+ R(RB)[0],

SUM[1]=R(RA)[aa]+ R(RB)[1],

SUM[2]=R(RA)[aa]+ R(RB)[2],

SUM[3]=R(RA)[aa]+ R(RB)[3].

또한, 판독 멀티플렉서(m0, m1, m2, m3)가 이미 존재하고 있기 때문에, 이 연산은 본래 자유롭다.

도 6(a) 내지 6(c)에 도시된 연산 외에, 도 1의 매트릭스(10)의 어레이의 데이터 요소들(즉, 로우 또는 서브컬럼)에 대한 선택도로 수행될 수 있는 많은 다른 연산이 있다. 이 선택도는 도 2 또는 도 4의 멀티플렉서(m0, m1, m2, m3)에 의해 제어된다.

도 7은 본 발명에 따른, 벡터 처리에 사용된 매트릭스의 로우 및 서브컬럼을 어드레싱하고, 어레이의 데이터 요소에 대한 선택도로 매트릭스의 어레이에 대한 연산을 수행하는 인스트럭션을 실행하는 프로세서(91)를 갖는 컴퓨터 시스템(90)을 도시한 것이다. 컴퓨터 시스템(90)은 프로세서(91)와, 프로세서(91)에 결합된 입력 장치(92)와, 프로세서(91)에 결합된 출력 장치(93)와, 프로세서(91)에 각각 결합된 메모리 장치(94, 95)를 포함한다. 프로세서(91)는 도 2 및 4의 프로세서(15)를 포함할 수도 있다. 입력 장치(92)는 특히 키보드, 마우스 등일 수도 있다. 출력 장치(93)는 특히 프린터, 플로터(plotter), 컴퓨터 스크린, 자기 테이프, 착탈식 하드디스크, 플로피디스크 등일 수도 있다. 메모리 장치(94, 95)는 특히, 하드디스크, 플로피 디스크, 자기 테이프, CD(compact disc) 또는 DVD(digital video disc)와 같은 광학 저장 장치, DRAM(dynamic random access memory), ROM(read-only memory) 등일 수도 있다. 메모리 장치(95)는 컴퓨터 코드(97)를 포함한다. 컴퓨터 코드(97)는 벡터 처리에 매트릭스의 로우 및 서브컬럼을 사용하고, 어레이의 데이터 요소에 대한 선택도로 매트릭스의 어레이에 대한 연산을 수행하는 인스트럭션을 실행하는 알고리즘을 포함한다. 프로세서(91)는 컴퓨터 코드(97)를 실행한다. 메모리 디바이스(94)는 입력 데이터(96)를 포함한다. 입력 데이터(96)는 컴퓨터 코드(97)에 의해 요구된 입력을 포함한다. 출력 장치(93)는 컴퓨터 코드(97)로부터의 출력을 디스플레이한다. 메모리 장치(94, 95)(또는 도 7에 도시되지 않은 하나 이상의 부가적인 메모리 장치) 중 하나 또는 둘 모두는 그 내부에 포함된 컴퓨터 판독가능한 프로그램 코드 및/또는 다른 데이터를 갖는 컴퓨터에 사용 가능한 매체(또는 컴퓨터 판독 가능 매체 또는 프로그램 저장 장치)로서 사용될 수도 있는데, 여기서 컴퓨터 판독 가능한 프로그램 코드는 컴퓨터 코드(97)를 포함한다. 일반적으로, 컴퓨터 시스템(90)의 컴퓨터 프로그램 제품은 상기 컴퓨터에 사용가능한 매체(또는 프로그램 저장 장치)를 포함할 수도 있다.

도 7은 하드웨어 및 소프트웨어의 특정 구성으로서 컴퓨터 시스템(90)을 도시하지만, 당해 분야에서 통상의 지식을 가진 자라면 알 수 있는 하드웨어 및 소프트웨어의 어떠한 구성도 도 7의 특정 컴퓨터 시스템(90)과 관련하여 위에서 언급한 목적을 위해 이용될 수도 있다. 예를 들면, 메모리 장치(94, 95)는 별도의 메모리 장치가 아니라 단일 메모리 장치의 일부분일 수도 있다.

이상, 예를 통해 본 발명의 실시예를 설명하였지만, 당업자에게는 자명한 많은 수정들 및 변형들이 이루어질 수도 있다. 따라서, 첨부한 청구범위는 본 발명의 사상 및 범주 내의 그러한 모든 수정 및 변형들을 포함하고자 한다.

발명의 효과

본 발명에 따르면 SIMD 벡터 처리에 사용된 매트릭스의 로우 및 컬럼을 어드레싱하기 위한 효과적인 프로세서 및 방법이 제공된다.

(57) 청구의 범위

청구항 1.

M 개의 독립적인 벡터 레지스터 파일을 포함하는 프로세서에 있어서,

상기 M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하도록 구성되고,

각각의 데이터 요소는 B 개의 2진 비트를 포함하며, 상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 포함하되, 상기 L=N*M이고,

각각의 컬럼은 K 개의 서브컬럼을 포함하되, 상기 N≥2, 상기 M≥2, 상기 K≥1, 상기 B≥1이며,

상기 N 개의 로우는 각각 어드레스가능하고,

상기 K 개의 서브컬럼은 각각 어드레스가능하며,

상기 프로세서는 상기 L 개의 데이터 요소를 중복 저장하지 않도록 구성되는

프로세서.

청구항 2.

제 1 항에 있어서,

상기 프로세서는 M 개의 어드레스 레지스터를 더 포함하고,

상기 M 개의 어드레스 레지스터는 각각 상기 M 개의 벡터 레지스터 파일 중 대응하는 하나의 벡터 레지스터 파일과 관련되며,

상기 M 개의 벡터 레지스터 파일 각각은 N 개의 레지스터의 어레이를 포함하고,

상기 M 개의 벡터 레지스터 파일의 상기 N*M 개의 레지스터 각각은 상기 L 개의 데이터 요소 중 하나의 데이터 요소를 저장하도록 구성되며,

각각의 벡터 레지스터 파일은 상기 벡터 레지스터 파일의 상기 N 개의 레지스터 중 하나를 지시하도록 구성되는 관련 어드레스 레지스터를 통해 독립적으로 어드레스 가능하며,

각각의 서브컬럼의 상기 데이터 요소는 상이한 벡터 레지스터 파일에 저장되도록 구성되고,

각각의 로우의 상기 데이터 요소는 상이한 벡터 레지스터 파일에 저장되도록 구성되는

프로세서.

청구항 3.

제 2 항에 있어서,

각각의 서브컬럼의 상기 데이터 요소는 상기 상이한 벡터 레지스터 파일의 상이한 상대적인 레지스터 위치에 저장되도록 구성되고,

각각의 로우의 상기 데이터 요소는 상기 상이한 벡터 레지스터 파일의 동일한 상대적인 레지스터 위치에 저장되도록 구성되는

프로세서.

청구항 4.

제 2 항에 있어서,

상기 프로세서는 상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 더 포함하고,

상기 매트릭스가 상기 M 개의 벡터 레지스터 파일에 저장되면,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 판독 로우 맵핑 알고리즘(a read-row mapping algorithm)에 따라서 상기 M 개의 벡터 레지스터 파일로부터의 상기 로우의 상기 데이터 요소를 상기 매트릭스의 상기 로우에 맵핑시킴으로써 상기 매트릭스의 로우를 판독하도록 구성되고,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 판독 서브컬럼 맵핑 알고리즘에 따라서 상기 M 개의 벡터 레지스터 파일로부터의 상기 서브컬럼의 상기 데이터 요소를 상기 매트릭스의 상기 서브컬럼으로 판독함으로써 상기 매트릭스의 서브컬럼을 판독하도록 구성되는

프로세서.

청구항 5.

제 2 항에 있어서,

상기 프로세서는 상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 더 포함하고,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 기록 로우 맵핑 알고리즘(a write-row mapping algorithm)에 따라서 상기 로우의 상기 데이터 요소를 상기 M 개의 벡터 레지스터 파일로 맵핑시킴으로써 상기 매트릭스의 로우를 기록하도록 구성되고,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 기록 서브컬럼 맵핑 알고리즘에 따라서 상기 서브컬럼의 상기 데이터 요소를 상기 M 개의 벡터 레지스터 파일로 맵핑시킴으로써 상기 매트릭스의 서브컬럼을 기록하도록 구성되는

프로세서.

청구항 6.

제 1 항에 있어서,

상기 프로세서는 M 개의 어드레스 레지스터를 더 포함하고,

상기 M 개의 어드레스 레지스터는 각각 상기 M 개의 벡터 레지스터 파일 중 대응하는 하나의 벡터 레지스터 파일과 관련되며,

상기 M 개의 벡터 레지스터 파일 각각은 N 개의 레지스터의 어레이를 포함하고,

상기 M 개의 벡터 레지스터 파일의 상기 N*M 개의 레지스터 각각은 상기 L 개의 데이터 요소 중 하나의 데이터 요소를 저장하도록 구성되며,

각각의 벡터 레지스터 파일은 상기 벡터 레지스터 파일의 상기 N 개의 레지스터 중 하나를 지시하도록 구성되는 관련 어드레스 레지스터를 통해 독립적으로 어드레스 가능하며,

상기 프로세서는 상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 더 포함하며, 상기 M 개의 멀티플렉서 각각은 상이한 값을 갖는

프로세서.

청구항 7.

매트릭스 데이터를 처리하기 위한 방법에 있어서,

프로세서를 제공하는 단계와,

상기 프로세서 내에 M 개의 독립적인 벡터 레지스터 파일을 제공하는 단계를 포함하되,

상기 M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하고,

각각의 데이터 요소는 B 개의 2진 비트를 포함하며,

상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 포함하되, 상기 L=N*M이고,

각각의 컬럼은 K 개의 서브컬럼을 포함하되, 상기 N≥2, 상기 M≥2, 상기 K≥1, 상기 B≥1이며,

상기 N 개의 로우는 각각 어드레스가능하고,

상기 K 개의 서브컬럼은 각각 어드레스가능하며,

상기 프로세서는 상기 L 개의 데이터 요소를 중복 저장하지 않는

매트릭스 데이터 처리 방법.

청구항 8.

제 7 항에 있어서,

상기 프로세서 내에 M 개의 어드레스 레지스터를 제공하는 단계를 더 포함하고,

상기 M 개의 어드레스 레지스터 각각은 상기 M 개의 벡터 레지스터 파일 중 대응하는 하나의 벡터 레지스터 파일과 관련되고,

상기 M 개의 벡터 레지스터 파일 각각은 N 개의 레지스터의 어레이를 포함하며,

상기 M 개의 벡터 레지스터 파일의 상기 N*M 개의 레지스터 각각은 상기 L 개의 데이터 요소 중 하나의 데이터 요소를 저장하고,

각각의 벡터 레지스터 파일은 상기 벡터 레지스터 파일의 상기 N 개의 레지스터 중 하나를 지시하도록 구성되는 관련 어드레스 레지스터를 통해 독립적으로 어드레스 가능하며,

각각의 서브컬럼의 상기 데이터 요소는 상이한 벡터 레지스터 파일에 저장되고,

각각의 로우의 상기 데이터 요소는 상이한 벡터 레지스터 파일에 저장되는

매트릭스 데이터 처리 방법.

청구항 9.

제 8 항에 있어서,

각각의 서브컬럼의 상기 데이터 요소는 상기 상이한 벡터 레지스터 파일의 상이한 상대적인 레지스터 위치에 저장되고,

각각의 로우의 상기 데이터 요소는 상기 상이한 벡터 레지스터 파일의 동일한 상대적인 레지스터 위치에 저장되는

매트릭스 데이터 처리 방법.

청구항 10.

제 8 항에 있어서,

상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 제공하는 단계를 더 포함하고,

상기 매트릭스가 상기 M 개의 벡터 레지스터 파일에 저장되면,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 판독 로우 맵핑 알고리즘에 따라서 상기 M 개의 벡터 레지스터 파일로부터의 상기 로우의 상기 데이터 요소를 상기 매트릭스의 상기 로우에 맵핑시킴으로써 상기 매트릭스의 로우를 판독하도록 구성되고,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 판독 서브컬럼 맵핑 알고리즘에 따라서 상기 M 개의 벡터 레지스터 파일로부터의 상기 서브컬럼의 상기 데이터 요소를 상기 매트릭스의 상기 서브컬럼으로 판독함으로써 상기 매트릭스의 서브컬럼을 판독하도록 구성되는

매트릭스 데이터 처리 방법.

청구항 11.

제 8 항에 있어서,

상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 제공하는 단계를 더 포함하고,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 기록 로우 맵핑 알고리즘에 따라서 상기 로우의 상기 데이터 요소를 상기 M 개의 벡터 레지스터 파일로 맵핑시킴으로써 상기 매트릭스의 로우를 기록하도록 구성되고,

상기 M 개의 멀티플렉서는 커맨드에 응답하여, 기록 서브컬럼 맵핑 알고리즘에 따라서 상기 서브컬럼의 상기 데이터 요소를 상기 M 개의 벡터 레지스터 파일로 맵핑시킴으로써 상기 매트릭스의 서브컬럼을 기록하도록 구성되는

매트릭스 데이터 처리 방법.

청구항 12.

제 7 항에 있어서,

상기 프로세서 내에 M 개의 어드레스 레지스터를 제공하는 단계를 더 포함하고,

상기 M 개의 어드레스 레지스터 각각은 상기 M 개의 벡터 레지스터 파일 중 대응하는 하나의 벡터 레지스터 파일과 관련되고,

상기 M 개의 벡터 레지스터 파일 각각은 N 개의 레지스터의 어레이를 포함하며,

상기 M 개의 벡터 레지스터 파일의 상기 N*M 개의 레지스터 각각은 상기 L 개의 데이터 요소 중 하나의 데이터 요소를 저장하고,

각각의 벡터 레지스터 파일은 상기 벡터 레지스터 파일의 상기 N 개의 레지스터 중 하나를 지시하도록 구성되는 관련 어드레스 레지스터를 통해 독립적으로 어드레스 가능하며,

상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 제공하여 상기 M 개의 멀티플렉서 각각이 상이한 값을 갖도록 하는 단계를 더 포함하는

매트릭스 데이터 처리 방법.

청구항 13.

제 7 항에 있어서,

상기 N 개의 로우 중 하나의 로우를 어드레싱하는 단계를 더 포함하는

매트릭스 데이터 처리 방법.

청구항 14.

제 7 항에 있어서,

K*M 개의 서브컬럼 중 하나의 서브컬럼을 어드레싱하는 단계를 더 포함하는

매트릭스 데이터 처리 방법.

청구항 15.

M 개의 독립적인 벡터 레지스터 파일을 포함하는 프로세서에 있어서,

상기 M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하도록 구성되고,

각각의 데이터 요소는 B 개의 2진 비트를 포함하며,

상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 포함하되, 상기 L=N*M이고,

각각의 컬럼은 K 개의 서브컬럼을 포함하되, 상기 N≥2, 상기 M≥2, 상기 K≥1, 상기 B≥1이며,

상기 N 개의 로우는 각각 어드레스가능하고,

상기 K 개의 서브컬럼은 각각 어드레스가능하며,

상기 매트릭스는 각각의 어레이가 상기 매트릭스의 로우 또는 서브컬럼인 어레이 세트를 포함하고,

상기 프로세서는 상기 어레이 세트의 제 1 어레이에 대한 연산을 수행하는 인스트럭션을 실행하도록 구성되며,

상기 연산은 상기 제 1 어레이의 상기 데이터 요소에 대한 선택도로 수행되는

프로세서.

청구항 16.

제 15 항에 있어서,

상기 프로세서는 상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 더 포함하고,

상기 M 개의 멀티플렉서와 관련된 값은 상기 선택도를 제어하는

프로세서.

청구항 17.

제 1 항 또는 15 항에 있어서,

상기 프로세서는 M 개의 어드레스 레지스터를 더 포함하고,

상기 M 개의 어드레스 레지스터는 각각 상기 M 개의 벡터 레지스터 파일 중 대응하는 하나의 벡터 레지스터 파일과 관련되며,

상기 M 개의 벡터 레지스터 파일 각각은 N 개의 레지스터의 어레이를 포함하고,

상기 M 개의 벡터 레지스터 파일의 상기 N*M 개의 레지스터 각각은 상기 L 개의 데이터 요소 중 하나의 데이터 요소를 저장하도록 구성되며,

각각의 벡터 레지스터 파일은 상기 벡터 레지스터 파일의 상기 N 개의 레지스터 중 하나를 지시하도록 구성되는 관련 어드레스 레지스터를 통해 독립적으로 어드레스 가능한

프로세서.

청구항 18.

제 15 항에 있어서,

상기 인스트럭션은 상기 어레이 세트의 상기 제 1 어레이의 적어도 하나의 데이터 요소를 상기 어레이 세트의 제 2 어레이에 복사하도록 구성되고,

상기 인스트럭션은 상기 제 1 어레이의 정확한 복사를 상기 제 2 어레이에 삽입하지 않는

프로세서.

청구항 19.

제 15 항에 있어서,

상기 인스트럭션은 상기 제 1 어레이 내의 상기 제 1 어레이의 상기 데이터 요소를 재배열하도록 구성되는

프로세서.

청구항 20.

제 15 항에 있어서,

상기 프로세서는 상기 L 개의 데이터 요소를 중복 저장하지 않도록 구성되는

프로세서.

청구항 21.

제 1 항 또는 15 항에 있어서,

L 개의 데이터 요소의 상기 매트릭스는 상기 M 개의 벡터 레지스터 파일에 저장되는

프로세서.

청구항 22.

매트릭스 데이터를 처리하기 위한 방법에 있어서,

프로세서를 제공하는 단계와,

상기 프로세서 내에 M 개의 독립적인 벡터 레지스터 파일을 제공하는 단계 -상기 M 개의 벡터 레지스터 파일은 L 개의 데이터 요소의 매트릭스를 집합적으로 저장하고, 각각의 데이터 요소는 B 개의 2진 비트를 포함하며, 상기 매트릭스는 N 개의 로우와 M 개의 컬럼을 포함하며, 상기 L=N*M이고, 각각의 컬럼은 K 개의 서브컬럼을 포함하며, 상기 N≥2, 상기 M≥2, 상기 K≥1, 상기 B≥1이며, 상기 N 개의 로우는 각각 어드레스가능하고, 상기 K 개의 서브컬럼은 각각 어드레스가능하며, 상기 매트릭스는 각각의 어레이가 상기 매트릭스의 로우 또는 서브컬럼인 어레이 세트를 포함함-와,

상기 프로세서에 의해 인스트럭션을 실행하는 단계 -상기 인스트럭션은 상기 어레이 세트의 제 1 어레이에 대한 연산을 수행하고, 상기 연산은 상기 제 1 어레이의 데이터 요소에 대한 선택도로 수행됨-

를 포함하는 매트릭스 데이터 처리 방법.

청구항 23.

제 22 항에 있어서,

상기 M 개의 벡터 레지스터 파일에 각각 결합된 M 개의 멀티플렉서를 제공하는 단계를 더 포함하고,

상기 M 개의 멀티플렉서와 관련된 값은 상기 선택도를 제어하는

매트릭스 데이터 처리 방법.

청구항 24.

제 7 항 또는 22 항에 있어서,

상기 프로세서 내에 M 개의 어드레스 레지스터를 제공하는 단계를 더 포함하고,

상기 M 개의 어드레스 레지스터 각각은 상기 M 개의 벡터 레지스터 파일 중 대응하는 하나의 벡터 레지스터 파일과 관련되고,

상기 M 개의 벡터 레지스터 파일 각각은 N 개의 레지스터의 어레이를 포함하며,

상기 M 개의 벡터 레지스터 파일의 상기 N*M 개의 레지스터 각각은 상기 L 개의 데이터 요소 중 하나의 데이터 요소를 저장하고,

각각의 벡터 레지스터 파일은 상기 벡터 레지스터 파일의 상기 N 개의 레지스터 중 하나를 지시하도록 구성되는 관련 어드레스 레지스터를 통해 독립적으로 어드레스 가능한

매트릭스 데이터 처리 방법

청구항 25.

제 22 항에 있어서,

상기 연산을 수행하는 단계는 상기 어레이 세트의 상기 제 1 어레이의 적어도 하나의 데이터 요소를 상기 어레이 세트의 제 2 어레이에 복사하는 단계를 포함하고,

상기 복사 단계는 상기 제 1 어레이의 정확한 복사를 상기 제 2 어레이에 삽입하지 않는

매트릭스 데이터 처리 방법.

청구항 26.

제 22 항에 있어서,

상기 연산을 수행하는 단계는 상기 제 1 어레이 내의 상기 제 1 어레이의 상기 데이터 요소를 재배열하는 단계를 포함하는

매트릭스 데이터 처리 방법.

청구항 27.

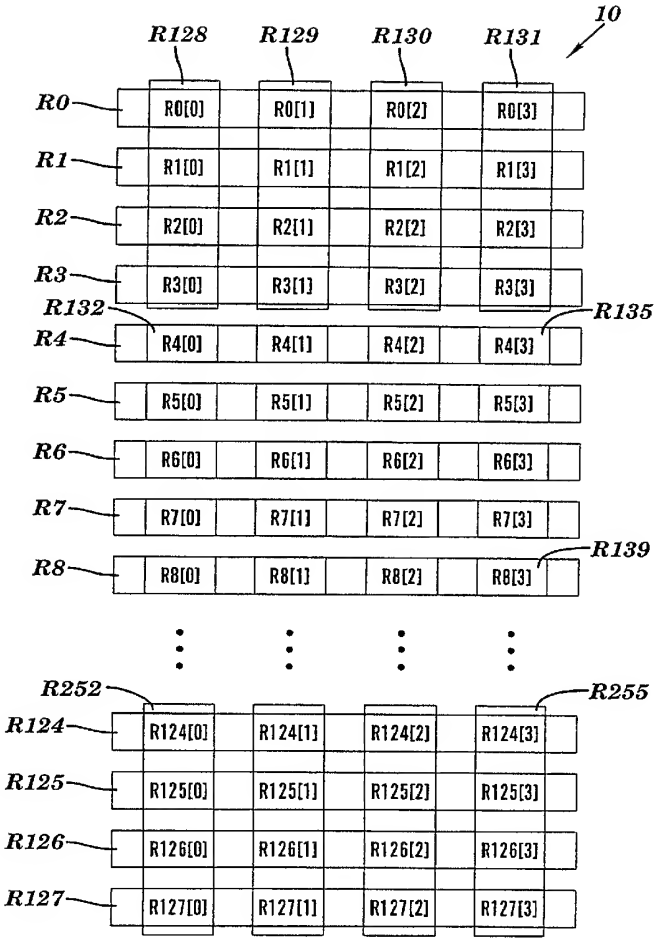
제 22 항에 있어서,

상기 프로세서는 상기 L 개의 데이터 요소를 중복 저장하지 않는

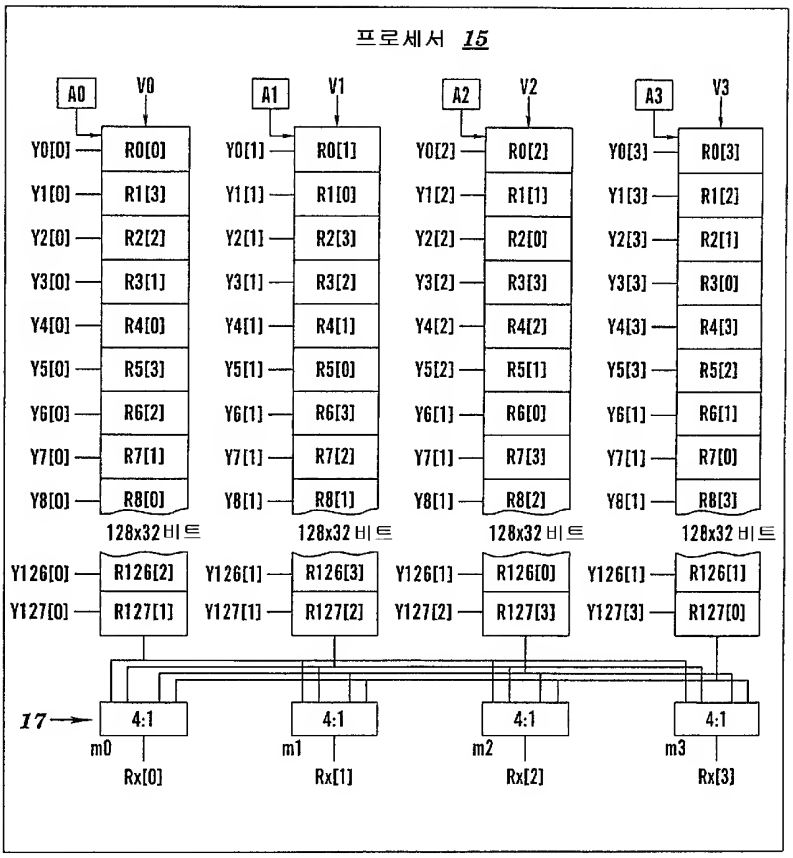
매트릭스 데이터 처리 방법.

도면

도면1


10
R128
R129
R130
R131
R0
R1
R2
R3
R132
R135
R4
R5
R6
R7
R139
R8
R252
R255
R124
R125
R126
R127
R0[0]
R0[1]
R0[2]
R0[3]
R1[0]
R1[1]
R1[2]
R1[3]
R2[0]
R2[1]
R2[2]
R2[3]
R3[0]
R3[1]
R3[2]
R3[3]
R4[0]
R4[1]
R4[2]
R4[3]
R5[0]
R5[1]
R5[2]
R5[3]
R6[0]
R6[1]
R6[2]
R6[3]
R7[0]
R7[1]
R7[2]
R7[3]
R8[0]
R8[1]
R8[2]
R8[3]
R124[0]
R124[1]
R124[2]
R124[3]
R125[0]
R125[1]
R125[2]
R125[3]
R126[0]
R126[1]
R126[2]
R126[3]
R127[0]
R127[1]
R127[2]
R127[3]

도면2

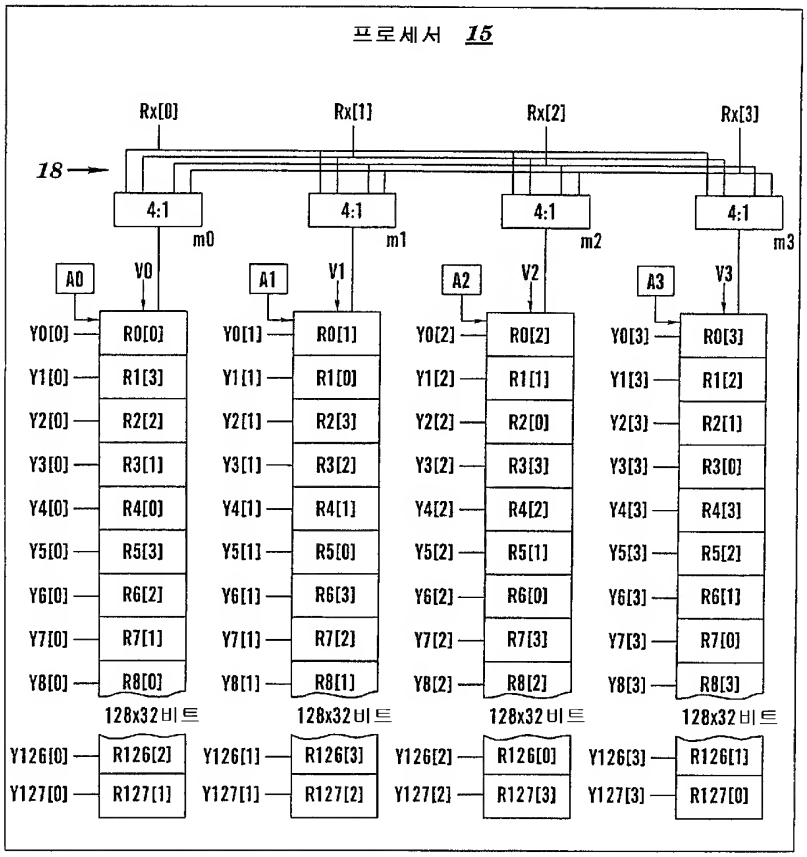

프로세서 15
A0
V0
A1
V1
A2
V2
A3
V3
Y0[0] R0[0]
Y1[0] R1[3]
Y2[0] R2[2]
Y3[0] R3[1]
Y4[0] R4[0]
Y5[0] R5[3]
Y6[0] R6[2]
Y7[0] R7[1]
Y8[0] R8[0]
Y0[1] R0[1]
Y1[1] R1[0]
Y2[1] R2[3]
Y3[1] R3[2]
Y4[1] R4[1]
Y5[1] R5[0]
Y6[1] R6[3]
Y7[1] R7[2]
Y8[1] R8[1]
Y0[2] R0[2]
Y1[2] R1[1]
Y2[2] R2[0]
Y3[2] R3[3]
Y4[2] R4[2]
Y5[2] R5[1]
Y6[1] R6[0]
Y7[1] R7[3]
Y8[1] R8[2]
Y0[3] R0[3]
Y1[3] R1[2]
Y2[3] R2[1]
Y3[3] R3[0]
Y4[3] R4[3]
Y5[3] R5[2]
Y6[1] R6[1]
Y7[1] R7[0]
Y8[1] R8[3]
128x32 비트
128x32 비트
128x32 비트
128x32 비트
Y126[0] R126[2]
Y127[0] R127[1]
Y126[1] R126[3]
Y127[1] R127[2]
Y126[1] R126[0]
Y127[2] R127[3]
Y126[1] R126[1]
Y127[3] R127[0]
17
4:1
4:1
4:1
4:1
m0
m1
m2
m3
Rx[0]
Rx[1]
Rx[2]
Rx[3]

도면3

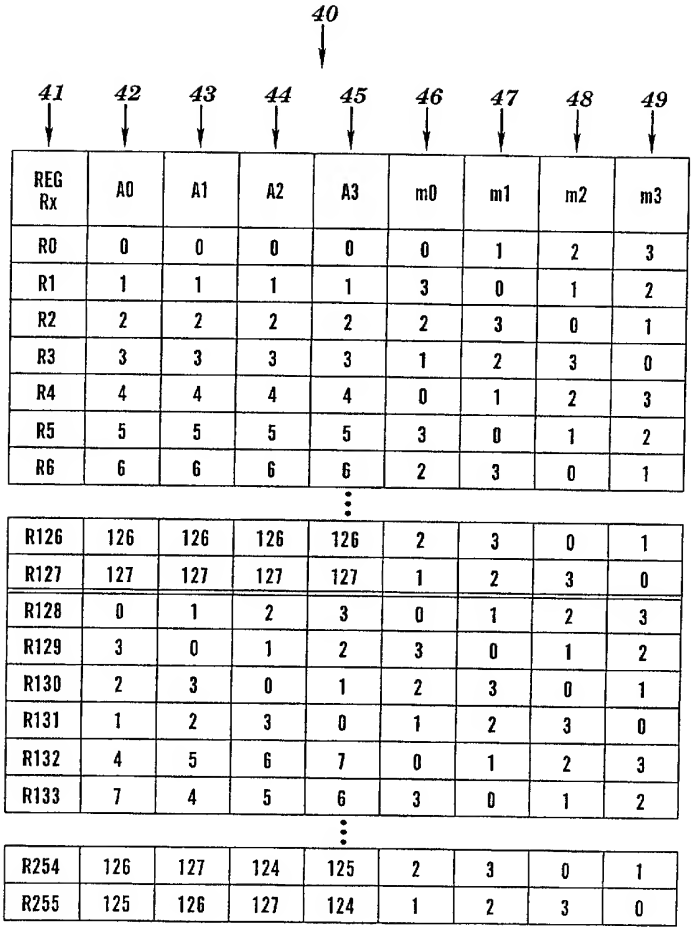
20

21	22	23	24	25	26	27	28	29
REG Rx	A0	A1	A2	A3	m0	m1	m2	m3
R0	0	0	0	0	0	1	2	3
R1	1	1	1	1	1	2	3	0
R2	2	2	2	2	2	3	0	1
R3	3	3	3	3	3	0	1	2
R4	4	4	4	4	0	1	2	3
R5	5	5	5	5	1	2	3	0
R6	6	6	6	6	2	3	0	1
⋮								
R126	126	126	126	126	2	3	0	1
R127	127	127	127	127	3	0	1	2
R128	0	1	2	3	0	1	2	3
R129	3	0	1	2	1	2	3	0
R130	2	3	0	1	2	3	0	1
R131	1	2	3	0	3	0	1	2
R132	4	5	6	7	0	1	2	3
R133	7	4	5	6	1	2	3	0
⋮								
R254	126	127	124	125	2	3	0	1
R255	125	126	127	124	3	0	1	2

도면4

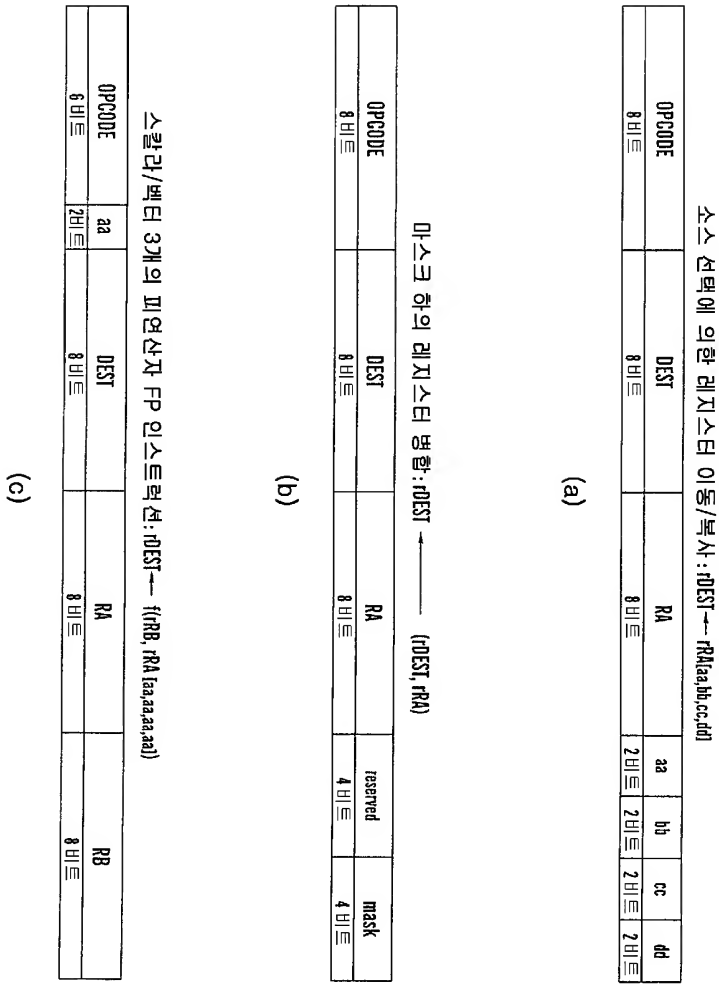

프로세서 15
Rx[0]
Rx[1]
Rx[2]
Rx[3]
18
4:1
m0
m1
m2
m3
A0
V0
A1
V1
A2
V2
A3
V3
Y0[0] R0[0]
Y1[0] R1[3]
Y2[0] R2[2]
Y3[0] R3[1]
Y4[0] R4[0]
Y5[0] R5[3]
Y6[0] R6[2]
Y7[0] R7[1]
Y8[0] R8[0]
Y0[1] R0[1]
Y1[1] R1[0]
Y2[1] R2[3]
Y3[1] R3[2]
Y4[1] R4[1]
Y5[1] R5[0]
Y6[1] R6[3]
Y7[1] R7[2]
Y8[1] R8[1]
Y0[2] R0[2]
Y1[2] R1[1]
Y2[2] R2[0]
Y3[2] R3[3]
Y4[2] R4[2]
Y5[2] R5[1]
Y6[2] R6[0]
Y7[2] R7[3]
Y8[2] R8[2]
Y0[3] R0[3]
Y1[3] R1[2]
Y2[3] R2[1]
Y3[3] R3[0]
Y4[3] R4[3]
Y5[3] R5[2]
Y6[3] R6[1]
Y7[3] R7[0]
Y8[3] R8[3]
128x32비트
Y126[0] R126[2]
Y127[0] R127[1]
Y126[1] R126[3]
Y127[1] R127[2]
Y126[2] R126[0]
Y127[2] R127[3]
Y126[3] R126[1]
Y127[3] R127[0]

도면5

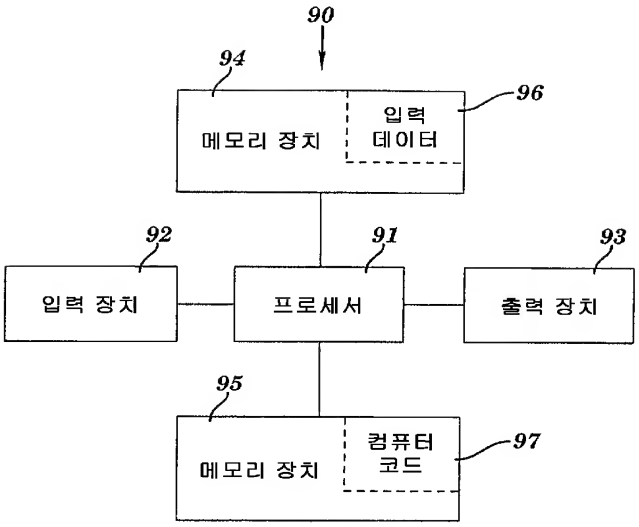


40	41	42	43	44	45	46	47	48	49
	REG Rx	A0	A1	A2	A3	m0	m1	m2	m3
	R0	0	0	0	0	0	1	2	3
	R1	1	1	1	1	3	0	1	2
	R2	2	2	2	2	2	3	0	1
	R3	3	3	3	3	1	2	3	0
	R4	4	4	4	4	0	1	2	3
	R5	5	5	5	5	3	0	1	2
	R6	6	6	6	6	2	3	0	1
	⋮								
	R126	126	126	126	126	2	3	0	1
	R127	127	127	127	127	1	2	3	0
	R128	0	1	2	3	0	1	2	3
	R129	3	0	1	2	3	0	1	2
	R130	2	3	0	1	2	3	0	1
	R131	1	2	3	0	1	2	3	0
	R132	4	5	6	7	0	1	2	3
	R133	7	4	5	6	3	0	1	2
	⋮								
	R254	126	127	124	125	2	3	0	1
	R255	125	126	127	124	1	2	3	0

도면6


(a)
소스 선택에 의한 레지스터 이동/복사 : rDEST ← rRA[aa,bb,cc,dd]
OPCODE 8비트
DEST 8비트
RA 8비트
aa 2비트
bb 2비트
cc 2비트
dd 2비트
(b)
마스크 하의 레지스터 병합 : rDEST ← (rDEST, rRA)
OPCODE 8비트
DEST 8비트
RA 8비트
reserved 4비트
mask 4비트
(c)
스칼라/벡터 3개의 피연산자 FP 인스트럭션 : rDEST ← f(rRB, rRA [aa,aa,aa,aa])
OPCODE 6비트
aa 2비트
DEST 8비트
RA 8비트
RB 8비트

도면7


90
94
메모리 장치
입력 데이터
96
92
입력 장치
91
프로세서
93
출력 장치
95
메모리 장치
컴퓨터 코드
97

(19) **KOREAN INTELLECTUAL PROPERTY OFFICE**

KOREAN PATENT ABSTRACTS

(11)Publication number: **1020050048465 A**
(43)Date of publication of application: **24.05.2005**

(21)Application number: **1020040083047**
(22)Date of filing: **18.10.2004**
(30)Priority: **18.11.2003 1**

(51)Int. Cl **G06F 9/30**

(71)Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**

(72)Inventor: **SANDON PETER A.**
WEST R. MICHAEL P.

(54) DEVICE AND METHOD FOR PROCESSING VECTOR REGISTER FILE

(57) Abstract:

PURPOSE: A device and a method for processing a vector register file are provided to logically address both rows and sub columns of a matrix stored in a plurality of vector register files within a processor. CONSTITUTION: The processor, comprising M independent vector register files, the M vector register files adapted to collectively store a matrix of L data elements, each data element having B binary bits, the matrix having N rows and M columns, the L=N*M, each column having K sub columns, the N>=2, the M>=2, the K>=1, the B>=1, each row of the N rows being addressable, each sub column of the K sub columns being addressable, the processor not adapted to duplicatively store the L data elements.

Legal Status
Date of request for an examination (20041018)
Notification date of refusal decision (00000000)
Final disposal of an application (registration)
Date of final disposal of an application (20060630)
Patent registration number (1006031240000)
Date of registration (20060712)
Number of opposition against the grant of a patent ( )
Date of opposition against the grant of a patent (00000000)
Number of trial against decision to refuse ( )
Date of requesting trial against decision to refuse ( )
Date of extinction of right ( )